Exercise 1:

Part(a)

Code:

```
clear all;
clc;
t = 0:0.001:2;
fa = 1; % frequency of a
Aa = 5; % amplitude of a
pa = 0; % phase of a
A = Aa*cos(2*pi*fa*t + pa);
figure(1)
subplot(2,2,1)
plot(t,A)
title('Continuous Time - Continuous Value Signal');
xlabel('Time (sec)');
ylabel('Signal A');
```

Explaination:

This code generates a continuous-time signal \(A \) over the time interval from 0 to 2 seconds. The signal is a cosine wave with a frequency of 1 Hz (cycles per second), an amplitude of 5, and zero phase. The 'subplot' function divides the figure window into a 2x2 grid and selects the first subplot to plot the signal. The resulting plot shows how the amplitude of the cosine wave varies with time. This code snippet is helpful for visualizing continuous signals in MATLAB, commonly encountered in fields such as signal processing and control systems.

Part(b)

Code:

```
n = 0:1:20;
fa = 0.1; % frequency of a
Aa = 5; % amplitude of a
pa = 0; % phase of a
A = Aa*cos(2*pi*fa*n + pa);
figure(1)
subplot(2,2,2)
stem(n,A)
title('Discrete Time - Continuous Value Signal');
xlabel('Time (sec)');
ylabel('Signal A');
```

Explanation:

This MATLAB code generates a discrete-time signal \(A \) over the range of integers from 0 to 20. The signal is a cosine wave with a frequency of 0.1 cycles per sample, an amplitude of 5, and zero phase. The 'stem' function is used to plot discrete points of the signal at integer time instances. The resulting plot shows how the amplitude of the cosine wave varies at each discrete time point. This code snippet is useful for understanding and visualizing discrete signals in MATLAB, which are commonly encountered in digital signal processing and communication systems.

Part (C):

Code

```
clear all;
clc;
t = 0:0.001:2;
fa = 1; % frequency of a
Aa = 5; % amplitude of a
pa = 0; % phase of a
A= Aa*cos(2*pi*fa*t + pa);
Y=round(A)
figure(1)
subplot(2,2,3)
plot(t,Y)
title('Continuous Time - Discrete Value Signal');
xlabel('Time (sec)');
ylabel('Signal A');
```

Explanation:

This MATLAB code generates a continuous-time sinusoidal signal \(A \) over the time interval from 0 to 2 seconds, then rounds each value to the nearest integer to create a discrete-value signal \(Y \). The 'subplot' function arranges the plots, and 'plot' visualizes the rounded signal over time. This process demonstrates how to convert a continuous signal into a discrete one by rounding its values, which can be useful for quantization in digital signal processing applications.

Part (C):

Code

```
clear all;
clc;
t = 0:1:20;
fa = 0.1; % frequency of a
Aa = 5; % amplitude of a
```

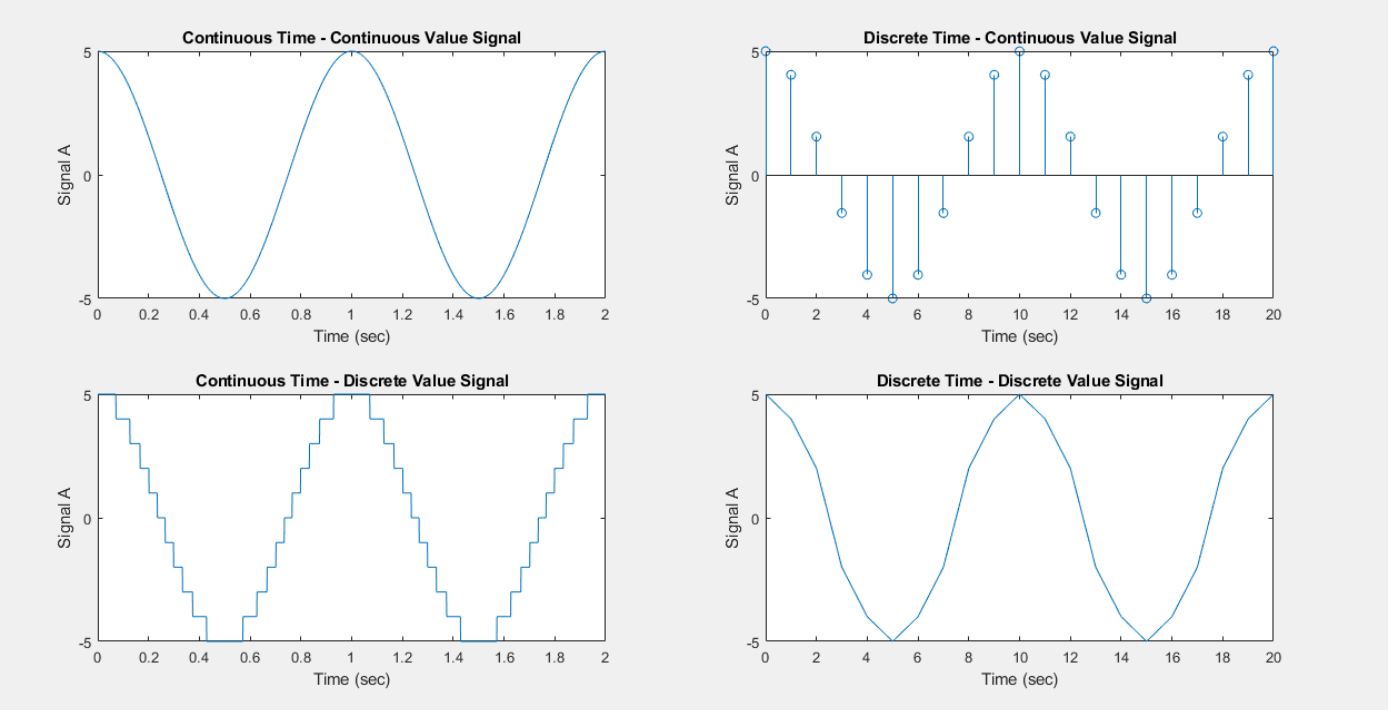
```
pa = 0; % phase of a
A= Aa*cos(2*pi*fa*t + pa);
Y=round(A)
figure(1)
subplot(2,2,4)
plot(t,Y)
title('Discrete Time - Discrete Value Signal');
xlabel('Time (sec)');
ylabel('Signal A');
```

Explanation:

This MATLAB code creates a discrete-time sinusoidal signal \(A \) over the time range from 0 to 20 seconds, then rounds each value to the nearest integer to produce a discrete-value signal \(Y \). The 'subplot' function arranges the plots, and 'plot' visualizes the rounded signal over discrete time instances. This process showcases how to convert a continuous sinusoidal signal into a discrete one with discrete values, which can be useful in digital signal processing applications where signals are represented digitally.

Exercise 1:

Output of a,b,c and d parts using subplot



❖ **Can a true continuous time or continuous value signal be defined in MATLAB? Why or why not?**

Ans:

In MATLAB, true continuous time or continuous value signals cannot be defined directly. This is because MATLAB operates in a digital environment where computations are performed on discrete data points. While MATLAB can simulate continuous-time behavior through techniques like using small time steps or interpolating between discrete points, it ultimately deals with discrete values. Real-world continuous signals are typically sampled and then processed in MATLAB as discrete-time signals. However, MATLAB provides functions and tools for working with discrete signals effectively, enabling simulations and analysis of continuous-time systems through techniques like numerical integration or Fourier analysis.

❖ **What is the difference between plot and stem?**

- Plot draws a smooth line connecting points, good for showing trends or continuous data.
- Stem shows individual points as markers on a vertical stem, useful for displaying discrete data or individual values without connecting lines.

Task 2 – Continuous Time convolution:

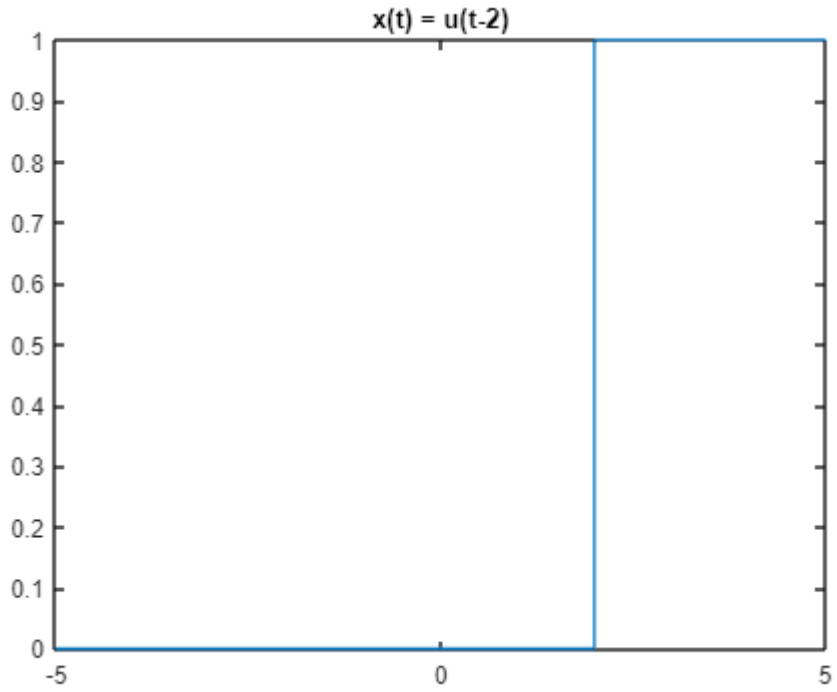
Exercise

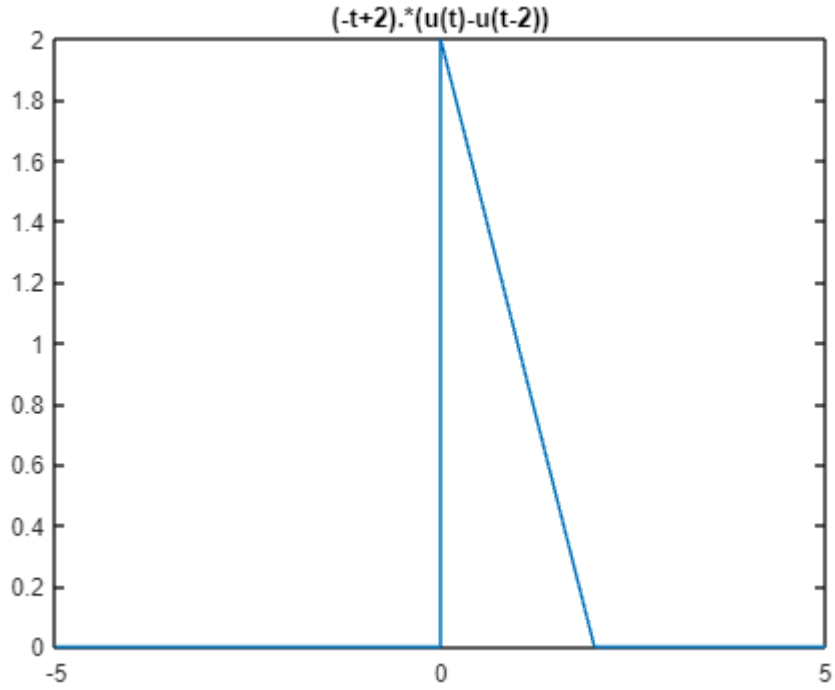
- ***a.***

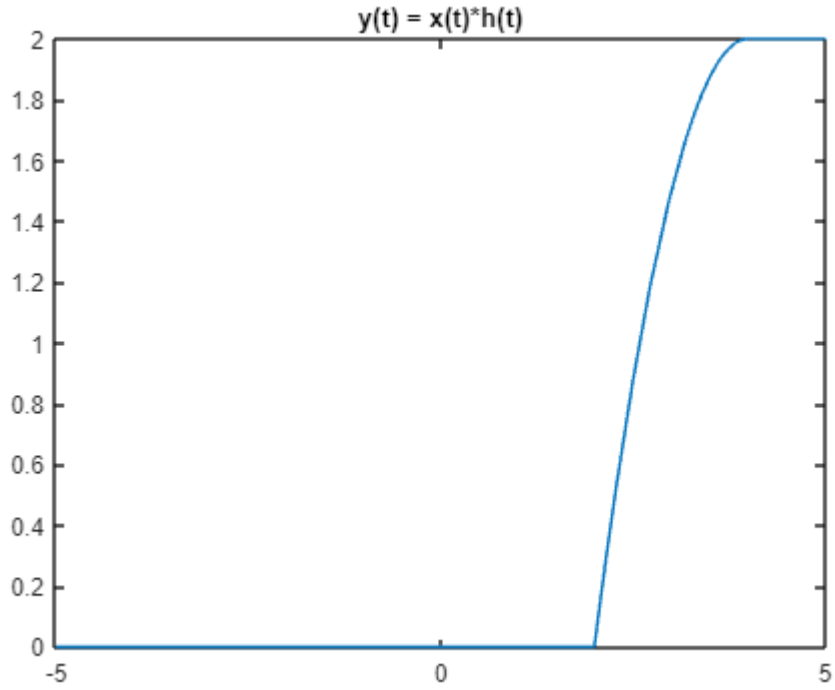
  Code:

```
syms w u(t) x(t) r(t) h(t) y(t)
u(t) = heaviside(t);
x(t) = u(t-2);
h(t) =  (-t+2).*(u(t)-u(t-2))
y(t) = int(x(w)*h(t-w), w, -inf, inf);
fplot(x(t))
title('x(t) = u(t-2)')
fplot(h(t))
title('(-t+2).*(u(t)-u(t-2))')
fplot(y(t))
title('y(t) = x(t)*h(t)')
```

  Output:







Explanation:

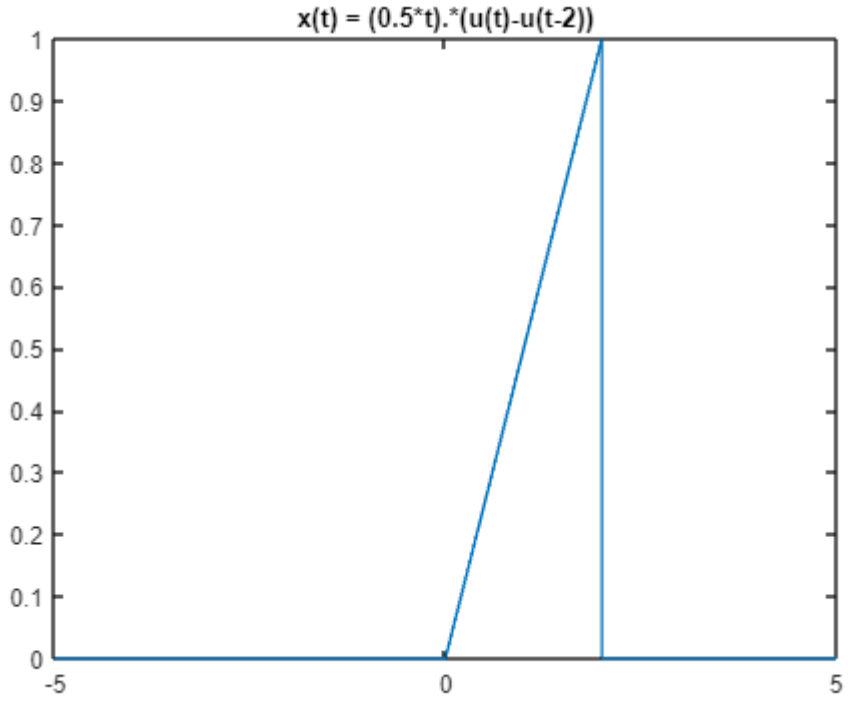
In this MATLAB code, we are defining symbolic functions and analyzing their behavior. We start by defining a step function u(t) that becomes 1 when t is greater than or equal to 0, and another function x(t) that represents a delayed step function by setting it to u(t-2). The function h(t) is defined as the product of (-t+2) and the difference of two step functions, creating a triangular pulse. Then, the convolution of x(t) and h(t) is calculated and stored in y(t). Finally, the code plots the graphs of x(t), h(t), and y(t) using the fplot function and provides titles for each plot. The first plot shows the delayed step function x(t), the second plot displays the triangular pulse h(t), and the third plot illustrates the convolution result y(t) of x(t) and h(t).

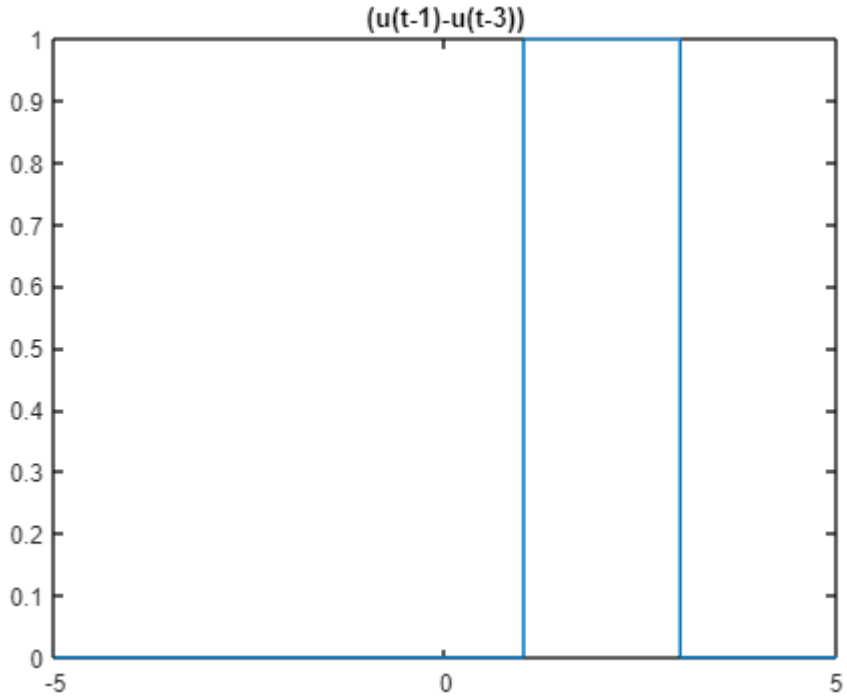
- ***b.***

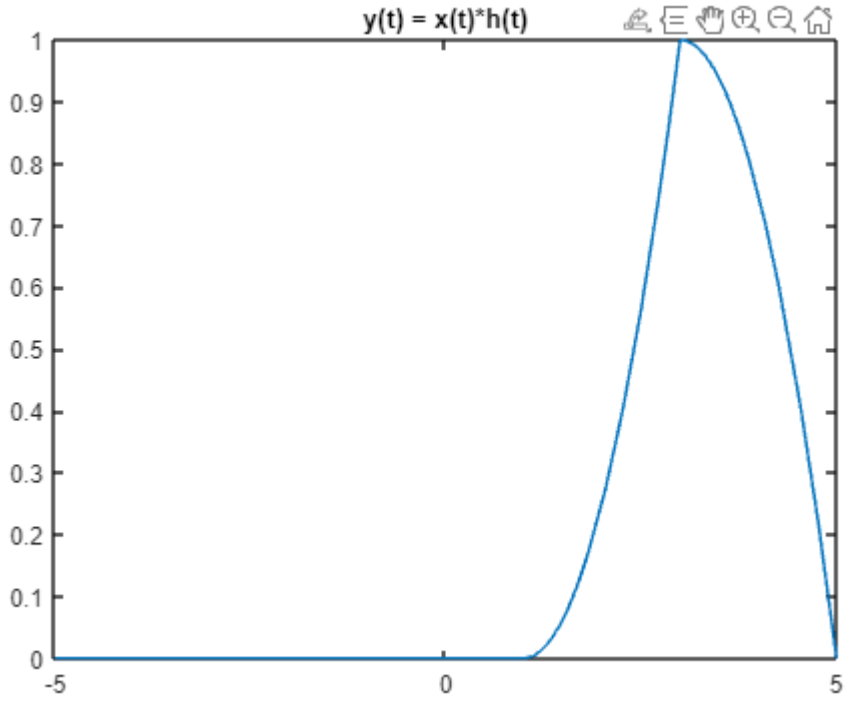
Code:

```
syms w u(t) x(t) r(t) h(t) y(t)
u(t) = heaviside(t);
x(t) = (0.5*t).*(u(t)-u(t-2));
h(t) =  (u(t-1)-u(t-3))
y(t) = int(x(w)*h(t-w), w, -inf, inf);
fplot(x(t))
title('x(t) = (0.5*t).*(u(t)-u(t-2))')
fplot(h(t))
title('(u(t-1)-u(t-3))')
fplot(y(t))
title('y(t) = x(t)*h(t)')
```

Output:







Explanation:

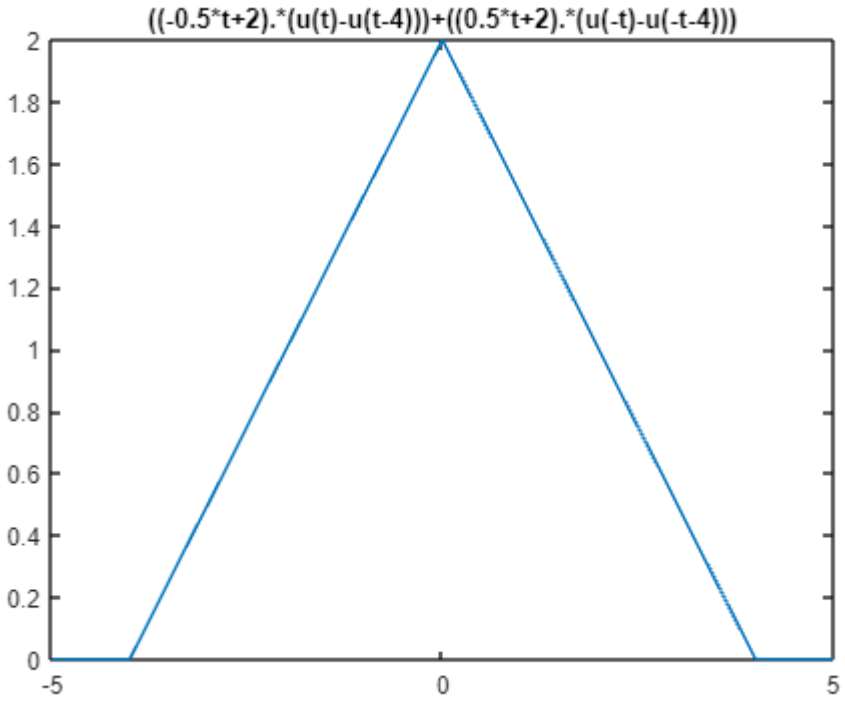
In this MATLAB code, symbolic functions are used to explore a signal processing scenario. The step function u(t) is defined, and then two other functions, x(t) and h(t), are created. The function x(t) represents a piecewise linear signal defined as 0.5*t for t between 0 and 2, and zero otherwise. The function h(t) is a rectangular pulse defined between t=1 and t=3. The code then calculates the convolution of x(t) and h(t), storing the result in y(t). The fplot function is used to graphically represent x(t), h(t), and y(t), each with their respective titles. The first plot depicts the piecewise linear signal x(t), the second plot illustrates the rectangular pulse h(t), and the third plot shows the convolution result y(t) of x(t) and h(t).

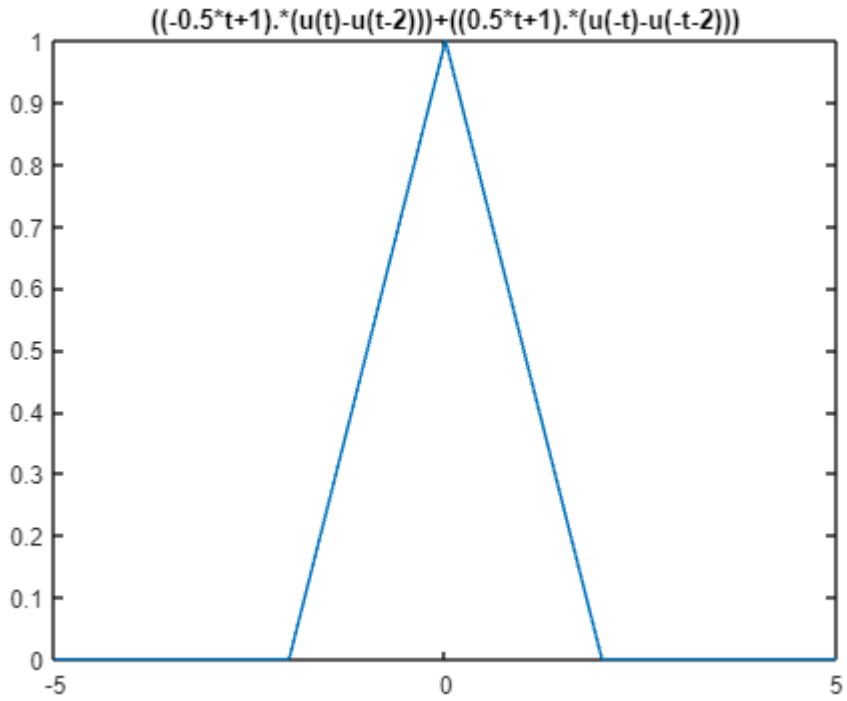
- ***c.***

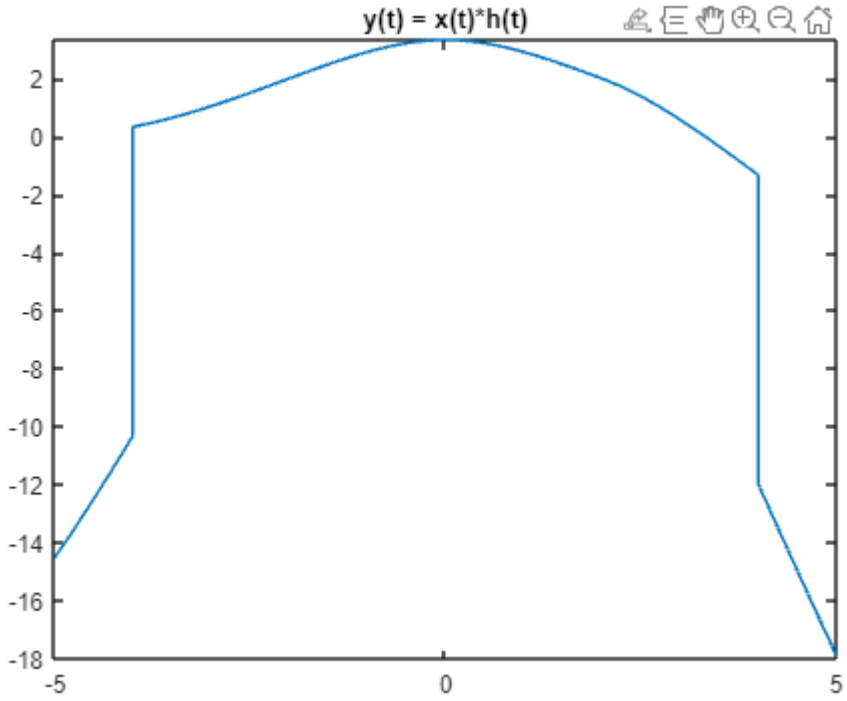
  Code:

```
syms w u(t) x(t) r(t) h(t) y(t)
u(t) = heaviside(t);
x(t) =((-0.5*t+2).*(u(t)-u(t-4)))+((0.5*t+2).*(u(-t)-u(-t-4)));
h(t) = ((-0.5*t+1).*(u(t)-u(t-2)))+((0.5*t+1).*(u(-t)-u(-t-2)));
y(t) = int(x(w)*h(t-w), w, -inf, inf);
fplot(x(t))
title('((-0.5*t+2).*(u(t)-u(t-4)))+((0.5*t+2).*(u(-t)-u(-t-4)))')
fplot(h(t))
title('((-0.5*t+1).*(u(t)-u(t-2)))+((0.5*t+1).*(u(-t)-u(-t-2)))')
fplot(y(t))
title('y(t) = x(t)*h(t)')
```

Output:







Explanation:

In this MATLAB code, symbolic functions are utilized to analyze a signal processing scenario. The step function u(t) is defined, and two functions, x(t) and h(t), are created. The function x(t) represents a piecewise linear signal composed of two segments: (-0.5t+2) for t between 0 and 4, and (0.5t+2) for t between -4 and 0. The function h(t) is also a piecewise linear signal with two segments: (-0.5t+1) for t between 0 and 2, and (0.5t+1) for t between -2 and 0. The code then calculates the convolution of x(t) and h(t), storing the result in y(t). The fplot function is employed to graphically represent x(t), h(t), and y(t), each with their respective titles. The first plot shows the piecewise linear signal x(t), the second plot illustrates the piecewise linear signal h(t), and the third plot displays the convolution result y(t) of x(t) and h(t).